Graph-Based Power Network Routing for Board-Level High Performance Systems

Rassul Bairamkulov and Eby G. Friedman
Department of Electrical and Computer Engineering
University of Rochester
Rochester, New York, USA
rbairamk@ur.rochester.edu, friedman@ece.rochester.edu

Abinash Roy, Mali Nagarajan, and Vaishnav Srinivas
Qualcomm Inc.
San Diego, California, USA
abinashr,mahaling,vaishnav@qti.qualcomm.com

***Abstract*—Unlike the routing of on-chip power delivery networks which is a highly automated process, the routing of board-level power nets is usually performed manually. The process is complicated by geometric and electrical constraints that impose restrictions on the routing process. An automated board-level power routing algorithm is presented here which provides efficient generation and refinement of power network geometries at the layout level. In a case study, a routing path connecting a power management integrated circuit to a ball grid array is routed using the automated tool, producing a low impedance network while complying with metal resource and geometric constraints.**

I. Introduction

The demand for high performance in modern integrated circuits requires reliable power delivery. Careful consideration of a variety of factors, such as voltage drops, temperature, and electromigration, is needed. Violations in power integrity may result in device malfunction or not satisfying target design specifications.

Whereas signal routing in integrated circuits (IC) is well developed and heavily automated, the design of the power delivery network at the board level is typically performed manually. A conventional design flow for a board-level power delivery network is shown in Fig. 1 [1]. The power delivery network of the package and board has historically been manually routed during or after cell placement and signal routing. During the physical design process, the controlled collapse chip connection (C4) bumps are assigned, which drives the placement of the ball grid array (BGA) pins of the package and printed circuit board (PCB). The design rules are concurrently set depending upon the system requirements. Any flaws in the electrical and thermal performance as well as manufacturability are identified during a later verification stage, and the system is iteratively modified to achieve the target requirements.

The power delivery design automation process has been previously discussed at various levels of abstraction [2], [3]. Most of the effort has concentrated on circuit-level analysis, specifically, power integrity [4], [5], [1], thermal performance [6], [7], and long term reliability [8], [9]. While highly regular grid structures are typically utilized in high performance integrated circuits [2], due to reliability and redundancy

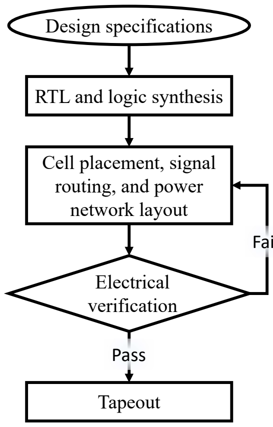

Fig. 1. Conventional design flow for power delivery networks [1].

considerations, the layout of the power delivery network at the board and package level is typically manually designed, connecting the output of the power management IC (PMIC) with the appropriate BGA balls, C4 bumps, and decoupling capacitors. Manual layout of the power delivery network requires significant resources in both labor and time [1]. This issue is further exacerbated by the iterative nature of the design process, demanding additional resources for modifying the power network, increasing the time to market.

Despite the relevance of the problem, automated generation of power delivery networks at the board level is rarely discussed in the literature. To the best of the authors' knowledge, synthesis of board-level power networks has not been discussed in the literature. In this paper, a novel graph-based automated layout generation methodology and related tool are presented. The automated tool produces one rail of a power delivery network in approximately four minutes. The tool, implemented in Python 3, minimizes the resistance and inductance of the resulting shapes while satisfying physical and electrical design rules, such as spacing and current density. The proposed algorithm provides a significant reduction in labor while greatly enhancing the efficiency of the design process of board-level power networks.

The rest of the paper is organized as follows. The power routing problem is discussed in section II. The proposed power routing algorithm is described in section III. The

This research is supported in part by the National Science Foundation under Grant No. CCF-1716091, IARPA under Grant No. W911NF-17-9-0001, and by grants from Cisco Systems, Qualcomm, Synopsys, and Google.

978-1-7281-3320-1/20/$31.00 ©2020 IEEE

automated power routing tool based on the proposed algorithm is demonstrated in section IV, followed by the summary and conclusions in section V.

II. Problem Statement

A typical layout of a PCB consists of several metal layers of different shape placed above each other. The metal layers are separated by a layer of dielectric and vias which connect to adjacent metal layers. Two sets of points, S and T, are identified, respectively, as the source and target locations. The objective of the routing algorithm is to produce a set of metal shapes and vias that minimizes the effective resistance and inductance between the source and target locations. Geometric, electrical, and design constraints also need to be considered during the routing process. Each metal layer may encounter different blockages, such as higher priority nets (e.g., signal nets, high performance processors), reserved whitespace, and ICs (on the top and bottom layers). Collectively, these blockages constitute geometric constraints. The metal segments and vias should be placed only in the remaining available space, referred to here as the routing space.

Many algorithms for interconnect routing in integrated circuits exist that are suitable for routing power nets [10], [11], [12], [13], [14]. Most of these algorithms employ a graph to yield the desired connections. Significant differences, however, exist between power and signal routing, as listed in Table I. The primary difference is the scale. Whereas the number of signal nets may often exceed thousands, the number of power nets is usually between 5 and 20, depending upon the system function, power consumption, cost, and form factor. The fundamental requirements for each net are, however, drastically different. Unlike signal nets, where the impedance characteristics are of lower priority, the electrical characteristics are pivotal to the power routing process, as the impedance directly affects the susceptance of the power network to resistive and inductive noise [3]. The design constraints in both problems are also different. While signal integrity issues are considered in signal routing, such as impedance matching, crosstalk, attenuation, inter-symbol interference, mode conversion, and timing, in power routing, the impedance, current density, and thermal behavior are of paramount importance.

III. Overview of Routing Algorithm

The process of power routing is depicted in Fig. 2 and described in this section. A method for determining the available space and conversion into a graph is described in subsection III-A. In the following sections, the subgraph construction procedure is described including an initial skeletal subgraph in subsection III-B, growth stage in subsection III-C, followed by a refinement stage in subsection III-D, and optional reheating stage in subsection III-E. The process is completed with back conversion of the graph into a layout, as described in subsection III-F.

A. Available routing space

The process of power routing commences with three important inputs. The net information describes which nets need to be routed. The layout template contains information describing the existing objects within the layout, including the signal traces, fucntional blocks, decoupling capacitors, inductors, and BGA balls. The design rules specify geometric constraints, such as the minimum space between objects.

The routing space for a given rail is initially the entire space of the PCB. Layout template objects foreign to the routed net are marked as blockages (see Fig. 2b). Areas within the proximity of these objects are removed from the routing space according to the design rules, characterizing a safe routing space for a given rail. The resulting space is a set of polygons A. Once the available space is determined, the space is divided into small individual cells. Each cell a is converted into a node of an equivalent graph $\mathit{\Gamma}$, and the adjacent cells are connected to the edges. To support back conversion into polygons, mapping between the node in graph $\mathit{\Gamma}$ and the cell location and shape is recorded as

$$f : \mathit{\Gamma} \rightarrow A, \tag{1}$$

$$f^{-1} : A \rightarrow \mathit{\Gamma}, \tag{2}$$

where f is the map converting a node within the graph $\mathit{\Gamma}$ into a polygonal cell, and f^{-1} is the inverse of f. The weight of each edge corresponds to the conductance between the corresponding cells. The cells in the equivalent graph disjoint from the source and target nodes are removed from the graph and not used in subsequent stages. Once the equivalent graph of the available space is constructed, the routing process becomes a graph-based optimization problem. For back conversion of the resulting subgraph into polygons, each node within the equivalent graph is associated with an underlying cell. Note that whereas most cells are a regular rectangular shape, cells adjacent to the blockages are often irregular in shape to avoid routing within prohibited areas.

With the construction of the equivalent graph, the routing objective is to find the subgraph $\mathit{\Gamma}_S$ within the available space graph $\mathit{\Gamma}$ that minimizes the resistance and loop inductance between the source and target nodes. Several optimization strategies exist that are applicable for this task [15]. A three stage graph-based procedure is proposed here. First, a skeletal route is created to provide an initial seed. Next, the subgraph growth stage is commenced where the area of the shape is increased until the target area is reached. Finally, the constructed shape is enhanced in the subgraph refinement stage, where the peripheral nodes are replaced with alternative nodes, improving the electrical characteristics of the routed shapes.

Two factors primarily affect the computational time of the routing process. First, the size of the subgraph $\mathit{\Gamma}_S$ is set by the granularity of the routing process. Fine granularity of the subgraph will likely result in lower impedance while increasing the time required for nodal analysis during the subgraph refinement process. Efficient sparse matrix solving algorithms are therefore utilized to accelerate this process. The number of refinement iterations is another parameter

TABLE I
COMPARISON BETWEEN SIGNAL AND POWER ROUTING

Feature	Signal Routing	Core Power Routing
Goal	Establish connection between signal source and destination locations	Establish connection between voltage source and ground locations
Resulting shape	Rectilinear metal tracks	Arbitrary shaped metal segment
Typical number of nets	More than 50	Less than 10
Constraints	Crosstalk noise, timing	Current density, temperature, metal resources

increasing the routing time. Additional iterations will likely lower the impedance at the cost of greater runtime.

B. Skeletal subgraph

The routing process requires an initial seed to be iteratively improved in later stages of the algorithm. The seed is generated in two steps. First, since there are no edges with negative conductance, the shortest path is guaranteed using the Dijkstra shortest path algorithm [16]. This skeletal subgraph, however, leaves any available space underutilized. Consider the situation shown in Fig. 2c. The shortest path skeletal subgraph is marked with a double line. If only the shortest path algorithm is used for the initial graph, the obstacle is bypassed from only one side, whereas a potentially better subgraph can be generated using a different skeleton.

To avoid this situation, the skeleton is reinforced using the biased random walk approach [17]. The initial node is set at the source node. The probability of transitioning toward the neighboring node depends upon the distance of these nodes from the target location,

$$P(x) = \frac{d(x)}{\sum_{i\in N(x_0)} d(i)}, \tag{3}$$

where $P(x)$ is the probability of transitioning to node x from node x_0, $d(x)$ is the Euclidean distance from node x to the target node, and $N(x_0)$ is the set of nodes adjacent to x_0. The set of nodes traveled by the random walk is used to construct the skeletal subgraph, which is appended to the shortest path graph (see Fig. 2c).

C. Growth stage

The resulting skeletal subgraph typically contains the fewest nodes, easily satisfying the area constraint. A procedure for adding nodes from the available space graph Γ to the resulting subgraph Γ_S is therefore effective. This problem is similar to the region growing problem [18], where the set of pixels within an image is grown into a cluster to identify specific patterns. The subgraph growing algorithm presented here, however, is drastically different due to the focus on improving the impedance characteristics of the graph. The set of peripheral nodes P_Γ is initially identified within the subgraph Γ_S. Neighbors of the nodes in P_Γ in the available space not belonging to the subgraph form the candidate node set C_Γ. The importance of the candidate node in C_Γ requires a robust heuristic to guide the creation of the conducting shape. A metric based on the current capacity of the adjacent nodes is proposed here.

Using the Laplacian matrix, the node voltages within the subgraph and the effective resistance between the source and target node are found using the following matrix equations,

$$R_{eff} = \boldsymbol{e_s}\boldsymbol{v}, \tag{4}$$

$$L\boldsymbol{v} = \boldsymbol{e_s}, \tag{5}$$

where s is the index of the source node, $\boldsymbol{e_s}$ is the vector equal to 1 at s and zero otherwise, and L and $\boldsymbol{v}$, are, respectively, the grounded Laplacian matrix and vector of node voltages. The current capacity of each peripheral node is

$$i_x = \sum_{k\in N(x)\wedge k\in\Gamma_S} |v_x - v_k|. \tag{6}$$

The resulting weight w_x of node x in C_Γ is

$$w_x = \sum_{k\in N(x)\wedge k\in\Gamma_S} i_x. \tag{7}$$

The candidate nodes are sorted and the nodes with the highest weights are appended onto the subgraph. The process is repeated until the target area of the shape is achieved.

D. Refinement stage

The subgraph resulting from the iterative growth process is typically suboptimal and may benefit from moving the nodes from areas with less current into areas where the current is more congested. The procedure is performed in four steps. First, the peripheral nodes conducting the smallest and largest currents are identified using (6) and (7) to form sets C and L,

$$L = \{w_l < w_{TL}, l\in N(\Gamma), l\in\Gamma_S, \Gamma\notin\Gamma_S\}, \tag{8}$$

$$C = \{w_c > w_{TH}, c\in N(a), a\in\Gamma_S, c\in\Gamma, c\notin\Gamma_S\}, \tag{9}$$

where w_{TL} and w_{TH} are, respectively, the lower and higher threshold weights. The set L is composed of the nodes in Γ_S conducting relatively small currents. The set C consists of nodes not belonging to the subgraph Γ_S, but adjacent to the nodes in Γ_S conducting large currents.

Next, the node from L with the lowest weight is removed from the subgraph Γ_S. After this step, the subgraph is checked for connections, since the removal of a node may create isolated islands. If a disconnect occurs due to the removal of a node, this node is eliminated from L and the process is repeated until a suitable node is found. Finally, the node with the largest weight in C is attached to the subgraph. This procedure is iteratively repeated, leading to a gradual improvement in the impedance characteristics of the subgraph.

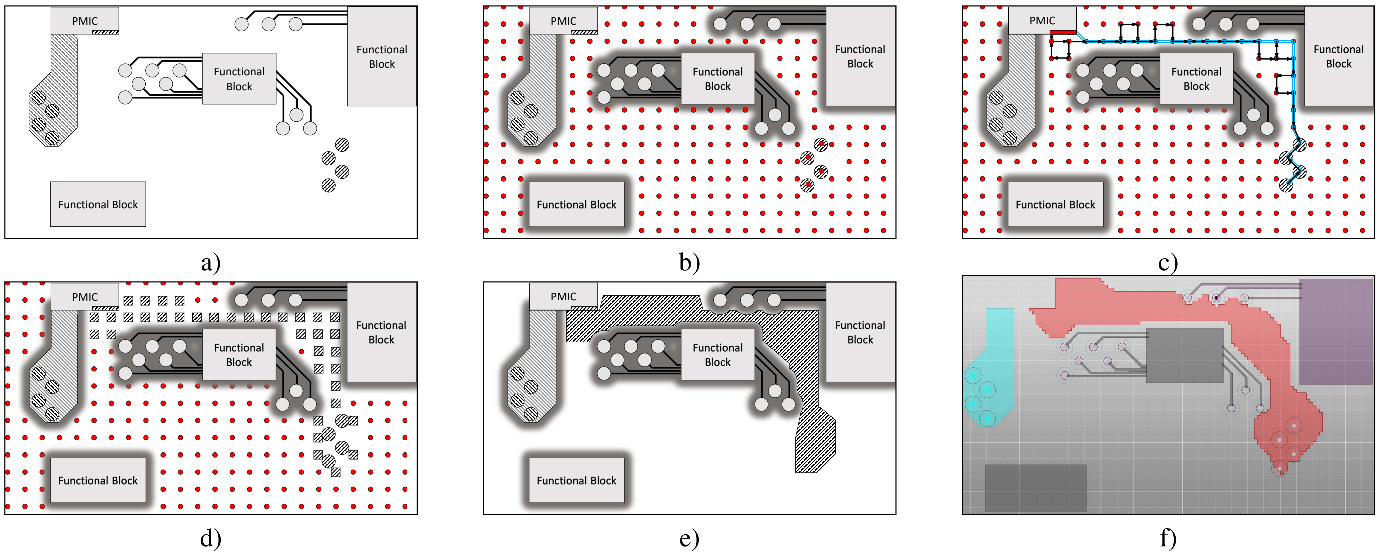

Fig. 2. Graph-based routing procedure. a) Initial topology. The signal via pads are depicted as gray circles, and the signal traces are shown as black lines. The existing power net is shaded from top-left to bottom-right. b) Generating the equivalent graph. The gray areas around the blocks, traces, and pads represent the necessary space to be maintained from the existing shapes. The red dots represent the graph vertices. For figure clarity, the edges connecting the adjacent vertices are not shown. c) Skeletal subgraph. The double line is the skeleton generated by the shortest path algorithm. The arrows represent the random walk used for an alternative skeletal graph. d) The square nodes represent the equivalent near-optimal subgraph. e) Final routed metal segment. f) Shape generated by the automated power routing algorithm.

To complete the refinement stage, an efficient termination strategy is required. A typical strategy in optimization algorithms, monitoring any change in the objective function, is suitable for this task. When the subgraph converges to a locally optimal state, any improvement in the impedance with each iteration reduces or becomes negative, i.e., the iteration produces a higher impedance. The refinement process may therefore be terminated if the change in the effective resistance is below a certain threshold.

E. Subgraph reheating

A common concern in global optimization is convergence of the algorithm into a local minimum [1]. To minimize the likelihood of this situation, the subgraph reheating procedure, adapted from simulated annealing [19], is proposed here. After termination of the iterative process, the nodes in Γ adjacent to the periphery of Γ_S are attached to the shape. This process is similar to the dilation operation in image processing [20], where a target image is expanded by adding neighboring pixels.

To restore the initial area of the subgraph, the process of peripheral node removal is performed, similar to the process described in subsection III-D. The importance of the peripheral nodes of the resulting dilated subgraph are evaluated using (6) and (7), and those nodes with the lowest weight are removed from the dilated subgraph. This process is iteratively repeated until the original target area of the shape is achieved.

F. Back conversion into polygons

Once the final topology of the subgraph is determined, mapping f, which is created during conversion of the available space A into the equivalent graph Γ, converts the subgraph Γ_S into the shape A_S. The aggregate of the subgraph cells is merged into the routed conducting shape, establishing the connection between the source, the PMIC, the target, and the BGA balls (see Fig. 2e). Practically, this stage is accomplished by utilizing modern open source geometry packages, such as Shapely [21] or Computational Geometry Algorithms Library [22].

IV. Case Study

To illustrate the power routing procedure, a demonstration PCB is designed in Ansys SIwave [23]. The topology of the PCB is based on Fig. 2a. The tool is implemented in Python 3 [24], and NetworkX [25] and Shapely [21] packages are used to manipulate, respectively, the graphs and polygons. A software interface has been developed to convert a NetworkX graph into a layout.

The dimensions of the routing space in the PCB are $1,302 \times 732$ units. The layout consists of three functional blocks, eleven signal BGA balls and corresponding traces, and one predefined power supply rail. The objective is to route the conducting shape between the output of the power management IC and the four BGA balls.

The target area of the resulting structure is $125,000$ units^2, and the graph cell size is 10 units $\times$ 10 units. For simplicity, electromigration and thermal constraints have not been considered in this case study. The algorithm successfully routes the shape from the PMIC to the BGA in 232 seconds, and the resulting shape is depicted in Fig. 2f. Note that the resulting shape is primarily composed of squares. The choice of this topology is to generate this shape from a large number of square cells, leading to a large number of right corners within the resulting polygon.

V. Conclusions

Power routing a PCB is a complicated process, drastically different from signal routing. The differences include greater flexibility in generating the shape and more priority on the impedance characteristics of the routed shape. A graph-based board-level power routing algorithm is presented in this paper where the routed shape is created in four steps. The routing space is initially determined by removing any blockages from the available area. The routing space is next converted into an equivalent graph. Third, subgraph growth and refinement are performed by generating a subgraph satisfying the area constraints, while improving the impedance characteristics. The subgraph reheating procedure may enhance the routing solution by minimizing the risk of converging into a local minimum. Finally, the resulting subgraph is converted into polygons for the physical layout. This procedure provides fast layout generation of power delivery networks, greatly reducing labor and time resources.

References

[1] R. Bairamkulov, K. Xu, M. Popovich, J. S. Ochoa, V. Srinivas, and E. G. Friedman, "Power Delivery Exploration Methodology based on Constrained Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019 (in press).

[2] E. Salman and E. G. Friedman, *High Performance Integrated Circuit Design*, McGraw-Hill Professional, 2012.

[3] I. Partin-Vaisband, R. Jakushokas, M. Popovich, A. V. Mezhiba, S. Köse, and E. G. Friedman, *On-Chip Power Delivery and Management, Fourth Edition*, Springer International Publishing, 2016.

[4] J. Mohamed, T. Michalka, S. Ozbayat, and G. R. Luevano, "PDN Design and Sensitivity Analysis Using Synthesized Models in DDR SI/PI Co-Simulations," *Proceedings of the IEEE Electrical Design of Advanced Packaging and Systems Symposium*, pp. 1–3, December 2018.

[5] H. Zhuang, W. Yu, S. Weng, I. Kang, J. Lin, X. Zhang, R. Coutts, and C. Cheng, "Simulation Algorithms with Exponential Integration for Time-Domain Analysis of Large-Scale Power Delivery Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 35, No. 10, pp. 1681–1694, October 2016.

[6] S. Mueller, A. K. Davis, M. L. F. Bellaredj, A. Singh, K. Z. Ahmed, M. Kar, S. Mukhopadhyay, P. A. Kohl, M. Swaminathan, Y. Wang, J. Wong, S. Bharathi, Y. Mano, A. Beece, B. Fasano, H. F. Moghadam, and D. Draper, "Modeling and Design of System-in-Package Integrated Voltage Regulator with Thermal Effects," *Proceedings of the IEEE Conference on Electrical Performance of Electronic Packaging and Systems*, pp. 65–68, October 2016.

[7] W. Zhu, G. Dong, Z. Mei, J. Zheng, J. Chai, and D. Song, "Modeling of Equivalent Thermal Conductivity of Power Delivery Network in 3D Packages," *Proceedings of the IEEE International Conference on Electronic Packaging Technology*, pp. 233–237, August 2018.

[8] S. Rahimipour, R. Zhang, K. Wang, K. Skadron, F. Z. B. Rokhani, and M. R. Stan, "MTTF Enhancement Power-C4 Bump Placement Optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 27, No. 7, pp. 1633–1639, July 2019.

[9] W. Chang, C. Lin, S. Mu, L. Chen, C. Tsai, Y. Chiu, and M. C. . Chao, "Generating Routing-Driven Power Distribution Networks with Machine-Learning Technique," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 36, No. 8, pp. 1237–1250, August 2017.

[10] V. F. Pavlidis and E. G. Friedman, "Timing-Driven Via Placement Heuristics for Three-Dimensional ICs," *Integration, The VLSI Journal*, Vol. 41, No. 4, pp. 489–508, July 2008.

[11] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott, and G. S. Taylor, "Magic: A VLSI Layout System," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 152–159, June 1984.

[12] M. Chen, K. G. Shin, and D. D. Kandlur, "Addressing, Routing, and Broadcasting in Hexagonal Mesh Multiprocessors," *IEEE Transactions on Computers*, Vol. 39, No. 1, pp. 10–18, January 1990.

[13] J. Wu, "A Fault-Tolerant and Deadlock-Free Routing Protocol in 2D Meshes based on Odd-Even Turn Model," *IEEE Transactions on Computers*, Vol. 52, No. 9, pp. 1154–1169, September 2003.

[14] M. Burstein and R. Pelavin, "Hierarchical Wire Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 2, No. 4, pp. 223–234, October 1983.

[15] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill Higher Education, 1994.

[16] S. Peyer, D. Rautenbach, and J. Vygen, "A Generalization of Dijkstra's Shortest Path Algorithm with Applications to VLSI Routing," *Journal of Discrete Algorithms*, Vol. 7, No. 4, pp. 377 – 390, December 2009.

[17] J. Guo, C. Zhang, and J. Cui, "Parallel Random Walk Algorithm in VLSI Analysis," *Proceedings of the International Conference on Computer Science and Electronics Engineering*, pp. 701–703, March 2013.

[18] A. Mehnert and P. Jackway, "An Improved Seeded Region Growing Algorithm," *Pattern Recognition Letters*, Vol. 18, No. 10, pp. 1065 – 1071, 1997.

[19] M. Lundy and A. Mees, "Convergence of an Annealing Algorithm," *Mathematical Programming*, Vol. 34, No. 1, pp. 111–124, January 1986.

[20] J. Y. Gil and R. Kimmel, "Efficient Dilation, Erosion, Opening, and Closing Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 12, pp. 1606–1617, December 2002.

[21] S. Gillies, "The Shapely User Manual," February 2018.

[22] H. Brönnimann, A. Fabri, G. Giezeman, S. Hert, M. Hoffmann, L. Kettner, S. Pion, and S. Schirra, "Computational Geometry Algorithms Library 4.14.1 - 2D and 3D Linear Geometry Kernel," September 2019.

[23] ANSYS, Inc., "ANSYS SIwave," 2015.

[24] G. van Rossum *et al.*, "The Python Language Reference: Release 3.8.0," October 2019.

[25] A. Hagberg, D. Schult, and P. Swart, "NetworkX Reference," October 2019.